

# Análise Comparativa de Arquiteturas de Deep Learning (Stacked LSTM vs. 1D-CNN) no Reconhecimento de Atividades Humanas

Seu Nome Completo

2025-07-12

## 1 Introdução

O monitoramento automático de atividades humanas (HAR - *Human Activity Recognition*) consolidou-se como um vetor de inovação na computação onipresente e saúde digital (Lara & Labrador, 2013). Sensores iniciais miniaturizados, integrados massivamente em *smartphones* e *wearables*, geram fluxos de dados contínuos que, quando processados por algoritmos inteligentes, permitem a inferência de comportamentos complexos.

A problemática central deste domínio reside na natureza estocástica, ruidosa e dependente do tempo das séries geradas pelo movimento humano. Enquanto métodos clássicos dependem de *feature engineering* manual, o Aprendizado Profundo (*Deep Learning*) oferece a capacidade de aprender representações hierárquicas diretamente dos dados brutos (LeCun et al., 2015).

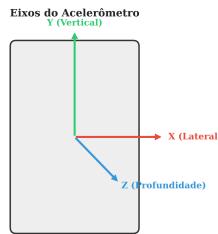
Este trabalho propõe um estudo comparativo rigoroso entre duas arquiteturas de redes neurais profundas: as **Redes Neurais Convolucionais Unidimensionais (1D-CNN)**, especializadas em extração de padrões locais (Kiranyaz et al., 2021), e as **Redes Neurais Recorrentes (Stacked LSTM)**, especializadas em dependências temporais (Hochreiter & Schmidhuber, 1997). O objetivo é avaliar a eficácia preditiva, a eficiência de parâmetros e as limitações físicas de cada topologia utilizando o *dataset* UCI HAR.

## 2 Fundamentos Teóricos e Metodológicos

### 2.1 Descrição e Natureza dos Dados (UCI HAR)

O estudo utiliza dados de 30 voluntários realizando Atividades de Vida Diária (ADL) com um *smartphone* na cintura. A amostragem ocorreu a 50Hz (Anguita et al., 2013).

**Figura 1: Sistema de Coordenadas do Smartphone para HAR**



Fonte: Adaptado de Anguita et al. (2013).

A entrada da rede é um tensor de dimensão  $(N, 128, 9)$ , onde  $N$  é o número de amostras, 128 são os passos de tempo (2,56s) e 9 são as variáveis dinâmicas: 1. **Aceleração Corporal** ( $Acc_{XYZ}^{body}$ ): 3 canais. Componente de movimento do usuário. 2. **Aceleração Total** ( $Acc_{XYZ}^{total}$ ): 3 canais. Inclui a gravidade, crucial para distinguir posturas estáticas. 3. **Velocidade Angular** ( $Gyro_{XYZ}$ ): 3 canais. Rotação do tronco.

## 2.2 Pré-processamento: Codificação One-Hot

Para viabilizar o treinamento supervisionado, os rótulos categóricos foram convertidos via **One-Hot Encoding**.

**Tabela 1:** Mapeamento de Classes e Codificação One-Hot

ID Original	Atividade	Índice Python	Vetor One-Hot (Target)
1	<b>Caminhando</b>	0	[1, 0, 0, 0, 0, 0]
2	<b>Subindo Escadas</b>	1	[0, 1, 0, 0, 0, 0]
3	<b>Descendo Escadas</b>	2	[0, 0, 1, 0, 0, 0]
4	<b>Sentado</b>	3	[0, 0, 0, 1, 0, 0]
5	<b>Em Pé</b>	4	[0, 0, 0, 0, 1, 0]
6	<b>Deitado</b>	5	[0, 0, 0, 0, 0, 1]

Fonte: Elaborada pelo autor.

## 2.3 Classificação Probabilística (Softmax)

A camada de saída utiliza a função **Softmax** para transformar os *logits*  $z$  em probabilidades normalizadas. A probabilidade  $P(y = i)$  da amostra pertencer à classe  $i$  é dada por:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Onde: \*  $z_i$ : Logit (saída linear) do neurônio correspondente à classe  $i$ . \*  $K$ : Número total de classes ( $K = 6$  neste estudo). \*  $e$ : Constante de Euler (base do logaritmo natural). \*  $\sum_{j=1}^K e^{z_j}$ : Fator de normalização que garante que a soma das probabilidades seja 1.

A predição final é a classe com maior probabilidade:  $\hat{y} = \text{argmax}(\sigma(z))$  (Goodfellow et al., 2016).

## 3 Modelagem Computacional e Arquiteturas

A implementação foi realizada via Keras/TensorFlow. Abaixo, detalha-se a configuração exata e o cálculo teórico de complexidade de cada modelo.

### 3.1 Arquitetura A: Stacked LSTM (Recorrente)

A rede utiliza duas camadas LSTM em série para extração profunda de tempo.

**Cálculo Teórico de Parâmetros:** A fórmula para LSTM é  $P = 4 \times [h \times (h + x + 1)]$ . \* **LSTM 1 (128 units):**  $4 \times [128 \times (128 + 9 + 1)] = 70.656$ . \* **LSTM 2 (64 units):**  $4 \times [64 \times (64 + 128 + 1)] = 49.408$ . \* **Total Estimado:** ~124k parâmetros.

**Tabela 2:** Resumo da Arquitetura Stacked LSTM

Camada	Configuração	Inicialização Pesos	Dropout Rate	Output Shape	Parâmetros
LSTM 1	128 units, return_seq=True	He Normal	-	(None, 128, 128)	70.656
Dropout 1	-	-	0.3	(None, 128, 128)	0
LSTM 2	64 units, return_seq=False	He Normal	-	(None, 64)	49.408

Camada	Configuração	Inicialização Pesos	Dropout Rate	Output Shape	Parâmetros
Dropout 2	-	-	0.3	(None, 64)	0
BatchNormalization		-	-	(None, 64)	256
Dense 1	64 units, ReLU	He Normal	0.2	(None, 64)	4.160
Output	6 units, Softmax	Glorot Uniform	-	(None, 6)	390
<b>Total</b>					<b>124.870</b>

Fonte: Elaborada pelo autor com dados do Keras.

### 3.2 Arquitetura B: Pure 1D-CNN (Convolucional)

Utiliza três blocos de convolução progressiva seguidos de *Global Average Pooling* (GAP).

**Cálculo Teórico de Parâmetros:** A fórmula para Conv1D é  $P = (k \times C_{in} \times F) + F$ . \* **Conv 1:**  $(3 \times 9 \times 64) + 64 = 1.792$ . \* **Conv 2:**  $(3 \times 64 \times 128) + 128 = 24.704$ . \* **Conv 3:**  $(3 \times 128 \times 256) + 256 = 98.560$ . \* **Total Estimado:** ~143k parâmetros.

**Tabela 3: Resumo da Arquitetura 1D-CNN**

Camada	Configuração	Inicialização Pesos	Dropout Rate	Output Shape	Parâmetros
Conv1D 1	64 filtros, k=3	He Normal	-	(None, 126, 64)	1.792
Conv1D 2	128 filtros, k=3	He Normal	-	(None, 61, 128)	24.704
Conv1D 3	256 filtros, k=3	He Normal	-	(None, 28, 256)	98.560
GAP	Global Average	-	0.5	(None, 256)	0
Dense 1	64 units, ReLU	He Normal	0.25	(None, 64)	16.448
Output	6 units, Softmax	Glorot Uniform	-	(None, 6)	390
<b>Total</b>	(Incluindo BNs)				<b>143.686</b>

Fonte: Elaborada pelo autor com dados do Keras.

### 3.3 Configuração de Hiperparâmetros e Treinamento

Para assegurar a reprodutibilidade e a robustez dos resultados, ambos os modelos foram submetidos a um protocolo de treinamento rigoroso, utilizando técnicas de regularização como Dropout (Srivastava et al., 2014) e Batch Normalization (Ioffe & Szegedy, 2015).

**Tabela 4: Hiperparâmetros e Estratégia de Otimização**

Parâmetro	Configuração	Justificativa Teórica
Otimizador	Adam	Momento adaptativo (Kingma & Ba, 2014).
Taxa de Aprendizado (LR)	0.001 (Inicial)	Padrão para convergência estável.

Parâmetro	Configuração	Justificativa Teórica
<b>Inicialização de Pesos</b>	He Normal	Otimizada para ReLU (He et al., 2015).
<b>Função de Perda</b>	Categorical Crossentropy	Métrica para classificação multiclasse.
<b>Critério de Parada</b>	Early Stopping (Patience=12)	Previne overfitting ao monitorar validação.
<b>Scheduler de LR</b>	ReduceLROnPlateau	Ajuste fino em platôs de erro.
<b>Batch Size</b>	64 amostras	Estabilidade estocástica do gradiente.

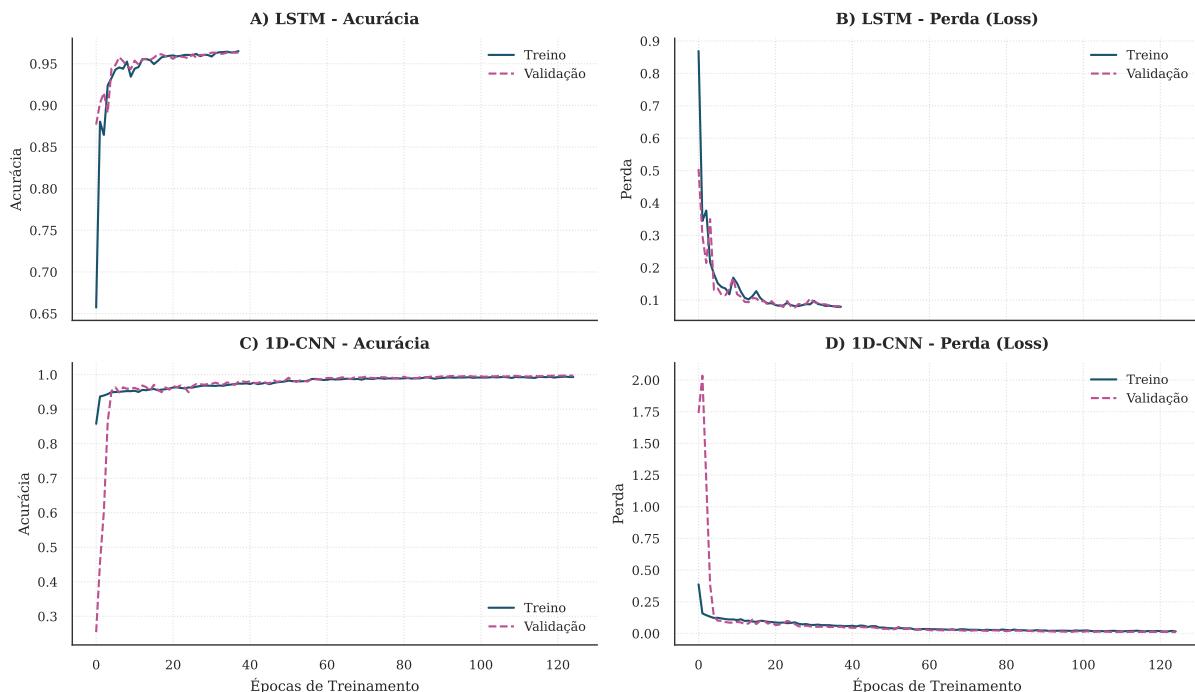
Fonte: Elaborada pelo autor.

## 4 Aplicação e Análise de Resultados

### 4.1 Dinâmica de Treinamento

A Figura 2 ilustra a convergência. A **1D-CNN (C-D)** converge mais rapidamente que a **LSTM (A-B)** nas primeiras épocas, estabilizando com uma acurácia elevada, enquanto a LSTM exige mais épocas para refinar as dependências temporais.

**Figura 2: Painel de Desempenho (Treino vs Validação)**



Legenda: A-B: Stacked LSTM; C-D: Pure 1D-CNN. Linhas sólidas: Treino; Tracejadas: Validação.

Fonte: Elaborada pelo autor.

### 4.2 Avaliação Comparativa (Dados de Teste)

A robustez foi verificada no conjunto de teste (2.947 amostras).

#### 4.2.1 Resultados: Stacked LSTM

O modelo recorrente atingiu uma acurácia global de **90%**.

**Tabela 5: Métricas Detalhadas - Stacked LSTM**

Classe	Precision	Recall	F1-Score	Suporte
<b>Caminhando</b>	0.98	0.92	0.95	496
<b>Subindo</b>	0.87	0.92	0.90	471
<b>Descendo</b>	0.86	1.00	0.92	420
<b>Sentado</b>	0.79	0.86	0.82	491
<b>Em Pé</b>	0.90	0.78	0.84	532
<b>Deitado</b>	<b>1.00</b>	<b>0.95</b>	<b>0.97</b>	537
<b>Média</b>	<b>0.90</b>	<b>0.90</b>	<b>0.90</b>	<b>2947</b>

Fonte: Elaborada pelo autor a partir dos resultados experimentais.

#### 4.2.2 Resultados: Pure 1D-CNN

O modelo convolucional superou o recorrente, atingindo uma acurácia global de **93%**.

**Tabela 6: Métricas Detalhadas - Pure 1D-CNN**

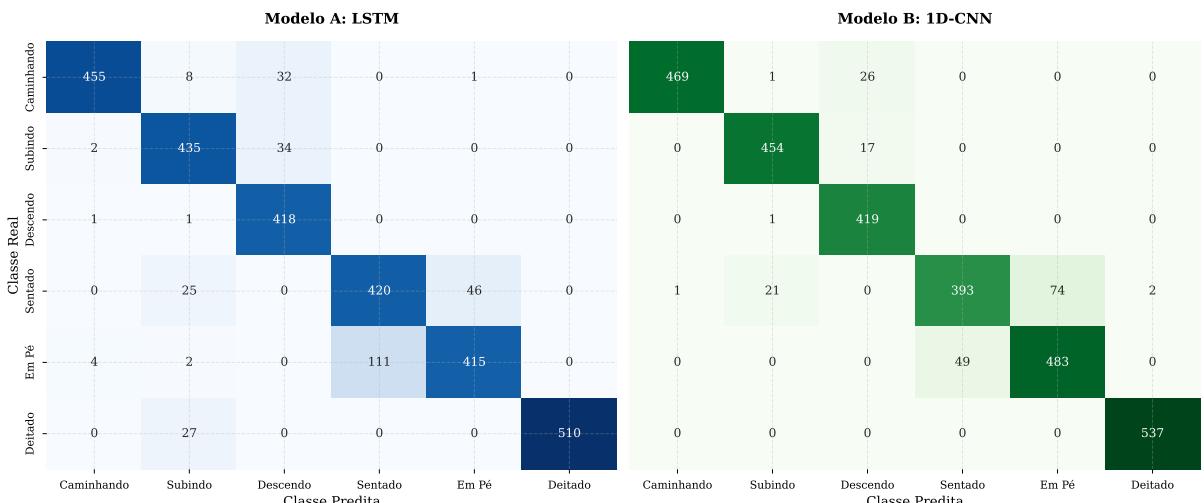
Classe	Precision	Recall	F1-Score	Suporte
<b>Caminhando</b>	1.00	0.95	0.97	496
<b>Subindo</b>	0.95	0.96	0.96	471
<b>Descendo</b>	0.91	1.00	0.95	420
<b>Sentado</b>	0.89	0.80	0.84	491
<b>Em Pé</b>	0.87	0.91	0.89	532
<b>Deitado</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	537
<b>Média</b>	<b>0.94</b>	<b>0.93</b>	<b>0.93</b>	<b>2947</b>

Fonte: Elaborada pelo autor a partir dos resultados experimentais.

### 4.3 Análise de Erros e Confusão

A análise das matrizes de confusão (Figura 3) revela as limitações físicas do sistema.

**Figura 3: Matrizes de Confusão Comparativas (Normalizadas)**



Legenda: Esquerda: Modelo LSTM; Direita: Modelo 1D-CNN.

Fonte: Elaborada pelo autor.

**Discussão dos Resultados:** 1. **Dinamismo:** Ambas as redes classificam com precisão quase perfeita movimentos cílicos (Caminhar, Subir/Descer Escadas). 2. **O Dilema Estático:** A maior fonte de erro, visível em ambas as matrizes, é a confusão entre “Sentado” e “Em Pé”. \* A LSTM teve maior dificuldade, com Precision de apenas 0.79 para “Sentado”. \* A CNN melhorou essa distinção (Precision 0.89), mas ainda confunde cerca de 20% das amostras. \* *Causa:* Em repouso, a aceleração total aproxima-se da gravidade ( $Acc \approx g$ ). Sem um barômetro para medir altitude, a orientação vetorial da gravidade é geometricamente idêntica nestas duas posturas (tronco ereto).

## 5 Conclusão

Este estudo comparou arquiteturas recorrentes e convolucionais para o reconhecimento de atividades humanas. A **1D-CNN** demonstrou superioridade sobre a **Stacked LSTM**, atingindo **93% de acurácia** (vs 90%) com uma convergência de treinamento mais estável e eficiente.

Apesar da CNN possuir um número ligeiramente maior de parâmetros totais (143k vs 124k), ela se beneficia da paralelização computacional e da capacidade de extrair assinaturas morfológicas locais (picos e formas de onda) que são altamente discriminativas em janelas curtas. A limitação persistente na distinção postural estática sugere a necessidade futura de fusão de sensores.

## 6 Referências

- ANGUITA, D. et al.** A Public Domain Dataset for Human Activity Recognition Using Smartphones. *ESANN 2013 proceedings*, Bruges, Belgium, 2013.
- GERS, F. A.; SCHMIDHUBER, J.; CUMMINGS, F.** Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2451-2471, 2000.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.** *Deep Learning*. MIT Press, 2016.
- HE, K. et al.** Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, 2015.
- HOCHREITER, S.; SCHMIDHUBER, J.** Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780, 1997.
- IOFFE, S.; SZEGEDY, C.** Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.
- KINGMA, D. P.; BA, J.** Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- KIRANYAZ, S. et al.** 1D Convolutional Neural Networks and Applications: A Survey. *Mechanical Systems and Signal Processing*, 151, 2021.
- LARA, O. D.; LABRADOR, M. A.** A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3), 1192-1209, 2013.
- LECUN, Y.; BENGIO, Y.; HINTON, G.** Deep learning. *Nature*, 521(7553), 436-444, 2015.
- SRIVASTAVA, N. et al.** Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929-1958, 2014.